UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/468,051	12/20/1999	THOMAS D. HARTNETT	RA-5271	3159

27516 7590 10/18/2004

UNISYS CORPORATION
MS 4773
PO BOX 64942
ST. PAUL, MN 55164-0942

EXAMINER
WOOD, WILLIAM H

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 10/18/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	**Application No.** 09/468,051	**Applicant(s)** HARTNETT ET AL.
	Examiner William H. Wood	**Art Unit** 2124

[illegible]

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 July 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-7 and 21-46* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-7 and 21-46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

Claims 1-7 and 21-46 are pending and have been examined.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) and the Intellectual Property and High Technology Technical Amendments Act of 2002 do not apply when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. Therefore, the prior art date of the reference is determined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

2. Claims 1, 3-4, 21, 27, 32, 39, 40 and 46 are rejected under 35 U.S.C. 102(e) as being anticipated by **Bhamidipati** et al. (USPN 6,112,295).

<u>*Claim 1*</u>

Bhamidipati claimed for use instruction processor that executes instructions included in a predetermined instruction set at an execution rate determined by a system clock signal *(column 2, lines 43-52)*, a synchronous instruction pipeline *(figure 2; column 2, lines 43-55; note one for each stage, synchronous)*, comprising:

- a pipeline execution circuit to process a first predetermined number of instructions simultaneously *(figure 1, elements 104 and 106; column 3, lines 58-61)*, each of said first predetermined number of instructions being in a respectively different stage of execution within said pipeline execution circuit *(column 1, lines 16-17)*, instructions being capable of advancing to a next stage of execution within said pipeline execution circuit at a time determined by the system clock signal *(column 1, lines 11-22; column 2, lines 43-52)*; and
- a pipeline fetch circuit coupled to provide each of the first predetermined number of instructions directly to said pipeline execution circuit *(figure 1, element 102, part of fetch circuit, directly coupled to element 104, part of execution circuit)*, the pipeline fetch circuit to retain a second predetermined number of instructions simultaneously (*column 1, lines 16-17)*, each of said second predetermined number of instructions being in a respectively different stage of processing within said pipeline fetch circuit *(figure 3, element 306; shown as own stage of processing)*, an instruction being capable of advancing to a next stage of execution within said pipeline fetch circuit at a time determined by the system clock signal and independently of the times at which instructions advance to a next

stage of execution within said pipeline execution circuit *(figure 3, element 306; column 3, lines 31-53; column 1, lines 24-35; decoupling queue allows stages before, fetch stages, to process regardless of the times or even whether stages after the queue, execution stages, are advancing; one decoupling queue implemented as the end of the last decode stage, see column 3, lines 57-58)*

Claim 3

Bhamidipati disclosed the synchronous instruction pipeline of Claim 1, wherein said pipeline fetch circuit includes a pre-decode logic circuit to generate pre-decode signals for an instruction that is in a pre-decode stage of processing within said pipeline fetch circuit, and wherein an instruction can enter said pre-decode stage of processing independently of the movement of instructions through said pipeline execution circuit *(figure 3, elements 304 and 306)*.

Claim 4

Bhamidipati disclosed the synchronous instruction pipeline of Claim 3, wherein said pipeline fetch circuit includes a decode logic circuit coupled to said pre-decode logic circuit to generate decode signals for an instruction that is in a decode stage of processing within said pipeline fetch circuit *(figure 1, element 102; figure 3, element 308)*, and wherein an instruction can enter said decode stage of processing from said pre-decode stage of processing independently of the movement of instructions through said pipeline execution circuit *(column 3, lines 54-58)*.

<u>Claim 21</u>

Bhamidipati disclosed for use in an instruction processor *(column 1, lines 6-16)* a synchronous pipeline circuit *(figure 2; column 2, lines 43-55; note one for each stage, synchronous)*, comprising:

- an execution circuit to provide a first predetermined number of execution stages *(figure 1, elements 104 and 106; column 3, lines 58-61)*, each being capable of performing a respective processing operation on a respective instruction *(column 1, lines 16-17)*;
- a fetch circuit coupled to the execution circuit to provide a second predetermined number of fetch stages *(figure 1, elements 100 and 102; figure 3, elements 300, 302, 304)*, each fetch stage being capable of performing a respective pre-execution operation on a respective instruction *(column 1, lines 16-17; column 2, lines 43-48)*, the fetch circuit to transfer each instruction processed by the fetch circuit directly from one of the fetch stages to one of the execution stages *(figure 1, element 102, part of fetch circuit, directly coupled to element 104, part of execution circuit,)* ones of the instructions processed within the fetch stages being capable of advancing to different available fetch stages *(as provided by pipelining concept, mentioned above)* independently of whether instructions are advancing within the execution stages *(figure*

3, element 306; column 3, lines 31-53; column 1, lines 24-35; decoupling queue allows stages before, fetch stages, to process regardless of the times or even whether stages after the queue, execution stages, are advancing; one decoupling queue implemented as the end of the last decode stage, see column 3, lines 57-58).

<u>*Claim 27*</u>

Bhamidipati disclosed a synchronous instruction pipeline circuit for processing instructions within a data processing system *(figure 2; column 2, lines 43-55; note one for each stage, synchronous)*, comprising:

- a first predetermined number of fetch stages to simultaneously process at least a first predetermined number of instructions *(figure 1, elements 100 and 102; figure 3, elements 300, 302, 304)*;
- a second predetermined number of execution stages to simultaneous process a second predetermined number of instructions *(figure 1, elements 104 and 106; column 3, lines 58-61)*, each received directly from one of the fetch stages *(figure 1, element 102, part of fetch circuit, directly coupled to element 104, part of execution circuit)*; and
- wherein at least one of the fetch stages is capable of providing an instruction to a different one of the fetch stages that is ready to receive an instruction irrespective of movement of instructions between the execution stages *(figure 3, element 306; column 3, lines 31-53; column 1, lines 24-35; decoupling queue allows stages before, fetch stages, to*

process regardless of the times or even whether stages after the queue, execution stages, are advancing; one decoupling queue implemented as the end of the last decode stage, see column 3, lines 57-58)

Claim 32

Bhamidipati disclosed a synchronous instruction pipeline to execute instructions *(figure 2; column 2, lines 43-55; note one for each stage, synchronous)*, comprising:

- an execution circuit having a first predetermined number of execution stages to execute a first predetermined number of instructions substantially simultaneously *(figure 1, elements 104 and 106; column 3, lines 58-61; column 1, lines 11-22)*; and
- a fetch circuit having a second predetermined number of fetch stages to perform pre-execution operations on at least a second predetermined number of instructions substantially simultaneously *(figure 1, elements 100 and 102; figure 3, elements 300, 302, 304; column 1, lines 16-17; column 2, lines 43-48)*, one of the fetch stages being coupled to provide each instruction processed by the fetch circuit directly to one of the execution stages *(figure 1, element 102, part of fetch circuit, directly coupled to element 104, part of execution circuit)*, at least one of the at least second predetermined number of instructions being capable of advancing between different ones of the

fetch stages regardless of whether an instruction is being transferred by the fetch circuit to the execution circuit *(figure 3, element 306; column 3, lines 31-53; column 1, lines 24-35; decoupling queue allows stages before, fetch stages, to process regardless of the times or even whether stages after the queue, execution stages, are advancing; one decoupling queue implemented as the end of the last decode stage, see column 3, lines 57-58)*.

Claim 39

Bhamidipati disclosed the pipeline of Claim 32, wherein the fetch circuit includes a circuit that allows instructions to advance within the second predetermined number of fetch stages if one of the execution stages is performing a predetermined function *(see claim 32; figure 3)*.

Claim 40

Bhamidipati disclosed a method of processing instructions within a synchronous pipeline of an instruction processor *(figure 2; column 2, lines 43-55; note one for each stage, synchronous)*, comprising:

a.) performing pre-execution operations on a first predetermined number of instructions substantially simultaneously within a first predetermined number of fetch stages of the pipeline *(figure 1, elements 100 and 102; figure 3, elements 300, 302, 304)*;

b.) executing a second predetermined number of instructions substantially simultaneously within a second predetermined number of execution stages of the pipeline *(figure 1, elements 104 and 106; column 3, lines 58-61; column 1, lines 11-22)*, wherein each of the second predetermined number of instructions were received directly from one of the fetch stages *(figure 1, element 102, part of fetch circuit, directly coupled to element 104, part of execution circuit)*; and

c.) allowing one or more of the first predetermined number of instructions to advance between ones of the fetch stages independently of whether any of the second predetermined number of instructions are advancing between ones of the execution stages *(figure 3, element 306; column 3, lines 31-53; column 1, lines 24-35; decoupling queue allows stages before, fetch stages, to process regardless of the times or even whether stages after the queue, execution stages, are advancing; one decoupling queue implemented as the end of the last decode stage, see column 3, lines 57-58)*.

Claim 46

Bhamidipati disclosed a pipeline circuit for use in an instruction processor *(column 1, lines 11-23; and column 2, lines 43-55)*, comprising:

- instruction fetch means for performing pre-execution operations on a first predetermined number of instructions substantially simultaneously within a fist predetermined number of fetch stages *(figure 1, elements 100 and 102; figure 3, elements 300, 302, 304)*;

- instruction execution means for executing a second predetermined number of instructions substantially simultaneously within a second predetermined number of execution stages *(figure 1, elements 104 and 106; column 3, lines 58-61; column 1, lines 11-22)*, each of the second predetermined number of instructions being received directly from one of the fetch stages *(figure 1, element 102, part of fetch circuit, directly coupled to element 104, part of execution circuit)*; and wherein
- the instruction fetch means includes means for allowing at least one of the first predetermined number of instructions to advance within the fetch stages irrespective of whether instructions are advancing within the execution stage *(figure 3, element 306; column 3, lines 31-53; column 1, lines 24-35; decoupling queue allows stages before, fetch stages, to process regardless of the times or even whether stages after the queue, execution stages, are advancing; one decoupling queue implemented as the end of the last decode stage, see column 3, lines 57-58)*.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to

> be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 2, 5-6, 22-25, 28-30, 33-38, 41-43 and 45 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bhamidipati** et al. (USPN 6,112,295) in view of **Kyker** et al. (USPN 6,026,477).

Claim 2

Bhamidipati did not explicitly state wherein said pipeline fetch circuit includes an instruction queue to store a predetermined maximum number of the instructions that are each ready to be processed by said pipeline fetch circuit. **Kyker** demonstrated that it was known at the time of invention to provide a queue to temporarily store an instruction from memory before heading to later fetch stages (column 1, lines 14-26). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati** pipeline with a prefetch queue as an intermediary between memory and the fetch stages as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to fetch as many instructions as possible to be waiting for processing at the earliest moment and in order to reduce idle time (**Kyker**: column 1, lines 14-16).

Claim 5

Bhamidipati and **Kyker** did not explicitly state the limitation wherein said pipeline fetch circuit includes a first selection circuit coupled to said pre-decode logic circuit to allow an instruction to be received by said pre-decode logic circuit at a time determined by the system clock signal if said decode logic circuit is available to accept an instruction currently being executed by said pre-decode logic circuit. **Kyker** demonstrated that it was known at the time of invention to for pipelines to stall (column 3, lines 40-45). A stall prevents a current stage from moving forward when the next stage is not available. It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati** and **Kyker**'s pipeline in such a manner as a pre-decode stage can only process instructions if clock allows it and the next stage (decode stage) will accept the previous pre-decode processed instruction as suggested by **Kyker**'s own teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated by the fact that stalls are common in pipeline technology (as indicated).

Claim 6

Bhamidipati and **Kyker** did not explicitly state the limitation wherein said pipeline fetch circuit includes a second selection circuit coupled to said decode logic circuit to allow an instruction to enter said decode stage of execution at a time determined by the system clock signal if said decode logic circuit is not processing another instruction. **Kyker** demonstrated that it was known at the time of invention to for pipelines to stall (column 3, lines 40-45). A stall prevents a current stage from moving forward when the next stage is not available. It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati** and **Kyker**'s pipeline in such a manner as a stage can only process instructions if clock allows it and the next stage will accept the previous processed instruction as suggested by **Kyker**'s own teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated by the fact that stalls are common in pipeline technology (as indicated).

Claim 22

Bhamidipati did not explicitly state wherein one of the fetch stages includes instruction address generate logic to predict which instructions are to enter the fetch stages. **Kyker** demonstrated that it was known at the time of invention to provide logic to predict instructions to fetch (column 2, lines 1-8). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline logic with prediction of instructions to fetch based upon branching as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to reduce idle time and avoid flushing incorrect instructions (**Kyker**: column 1, lines 18-22 and column 2, lines 61-66).

Claim 23

Bhamidipati and **Kyker** further disclosed the limitation wherein the instruction address generate logic includes a circuit to clear ones of the fetch stages in response to a determination that instruction execution was re-directed *(**Kyker**: column 2, lines 54-66).*

Claim 24

Bhamidipati disclosed the pipeline circuit of Claim 21, and further including

- including a memory to store instructions *(column 2, lines 30-55)*;

Bhamidipati did not explicitly state the limitations:

- a queue coupled to the memory to temporarily store at least one instruction fetched from the memory; and
- a circuit coupled to the queue and to at least one of the fetch stages to fetch an instruction from the queue for presentation to at least one of the fetch stages.

Kyker demonstrated that it was known at the time of invention to provide a queue to temporarily store an instruction from memory and a circuit coupled to the queue and a fetch stage to fetch instructions from the queue to the fetch stages (column 1, lines 14-26). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline with a prefetch queue as an intermediary between memory and the fetch stages as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to fetch as many instructions as possible to be waiting for processing at the earliest moment and avoid idle time (**Kyker**: column 1, lines 14-26).

Claim 25

Bhamidipati and **Kyker** further disclosed the limitation wherein the circuit coupled to the queue is capable of retrieving instructions from the queue for presentation to the at least one of the fetch stages regardless of whether instructions are advancing within the execution stages *(**Bhamidipati**'s decoupling queue system Figure 3, element 306 allows the circuit to remain independent of the execution stages, as previously indicated).*

Claim 28

Bhamidipati did not explicitly state the pipeline circuit of Claim 27, wherein one of the fetch stages includes address generate logic to predict which instructions are to enter the fetch stages. **Kyker** demonstrated that it was known at the time of invention to provide logic to predict instructions to fetch (column 2, lines 1-8). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline logic with prediction of instructions to fetch based upon branching as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to reduce idle time and avoid flushing incorrect instructions (**Kyker**: column 1, lines 18-22 and column 2, lines 61-66).

Claim 29

Bhamidipati and **Kyker** further disclosed the limitation wherein the address generate logic includes a circuit to flush one or more instructions from the fetch stages if it is determined that a misprediction occurred *(**Kyker**: column 2, lines 54-66).*

Claim 30

Bhamidipati disclosed the pipeline circuit of Claim 27, and further including:

- a memory *(column 2, lines 30-55)*; and

Bhamidipati did not explicitly state:

- a storage device coupled to one of the fetch stages and to the memory to store instructions retrieved from the memory, wherein a predetermined number of instructions may be stored within the storage device regardless of whether instructions are advancing within the fetch stages.

Kyker demonstrated that it was known at the time of invention to provide a queue to temporarily store an instruction from memory and a circuit coupled to the queue and a fetch stage to fetch instructions from the queue to the fetch stages (column 1, lines 14-26). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline with a prefetch queue as an intermediary between memory and the fetch stages as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to fetch as many instructions as possible to be waiting for processing at the earliest moment and avoid idle time (**Kyker**: column 1, lines 14-26).

Claim 33

Bhamidipati did not explicitly state the pipeline of Claim 32, wherein one of the fetch stages includes instruction address generate logic to determine which instructions are to enter the fetch circuit. **Kyker** demonstrated that it was known at the time of invention to provide logic to predict instructions to fetch (column 2, lines 1-8). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline logic with prediction of instructions to fetch based upon branching as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to reduce idle time and avoid flushing incorrect instructions (**Kyker**: column 1, lines 18-22 and column 2, lines 61-66).

Claim 34

Bhamidipati and **Kyker** disclosed the limitation wherein the instruction address generate section includes a circuit to remove instructions from the fetch circuit during a pipeline flush operation (**Kyker**: column 2, lines 54-64).

Claim 35

Bhamidipati disclosed the pipeline of Claim 32, and further including:

- a memory to store instructions *(column 2, lines 30-55)*; and

Bhamidipati did not explicitly state:

- a queue coupled to store instructions from the memory, the queue further being coupled to provide an instruction to the fetch circuit if one of the fetch stages is available and irrespective of whether an instruction is being provided from the fetch circuit to the execution circuit.

Kyker demonstrated that it was known at the time of invention to provide a queue to temporarily store an instruction from memory and a circuit coupled to the queue and a fetch stage to fetch instructions from the queue to the fetch stages (column 1, lines 14-26). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline with a prefetch queue as an intermediary between memory and the fetch stages as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to fetch as many instructions as possible to be waiting for processing at the earliest moment and avoid idle time (**Kyker**: column 1, lines 14-26).

<u>*Claim 36*</u>

Bhamidipati and **Kyker** disclosed a circuit coupled to the queue to allow an instruction to be stored to the queue independently of whether an instruction is advancing within the fetch circuit *(**Kyker**: column 1, lines 14-26)*.

<u>Claim 37</u>

Bhamidipati and **Kyker** disclosed the limitation wherein the circuit allows a predetermined maximum number of instructions to be stored to the queue independently of whether an instruction is advancing within the fetch circuit *(**Kyker**: column 1, lines 14-26).*

<u>Claim 38</u>

Bhamidipati and **Kyker** did not explicitly state the limitation wherein the one of the fetch stages includes a circuit to allow retrieval of an instruction from either the memory or from the queue. **Kyker** demonstrated that it was known at the time of invention to flush a pipeline (column 2, lines 54-66). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati** and **Kyker**'s pipeline with the ability to choose between fetching instructions directly from memory or from a prefetch queue as needed as suggested by **Kyker**'s teachings on pipeline flush. This implementation would have been obvious because one of ordinary skill in the art would be motivated to retrieve instructions from the queue if they existed and from memory directly if the queue had had to be emptied (from memory is the fastest).

<u>Claim 41</u>

Bhamidipati disclosed the limitations:

- fetching an instruction from a memory *(column 2, lines 30-55)*;

Bhamidipati did not explicitly state:

- storing the instruction within a queue; and
- retrieving the instruction from the queue to undergo a pre-execution operation within a predetermined one of the fetch stages

Kyker demonstrated that it was known at the time of invention to provide a queue to temporarily store an instruction from memory and a circuit coupled to the queue and a fetch stage to fetch instructions from the queue to the fetch stages (column 1, lines 14-26). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s pipeline with a prefetch queue as an intermediary between memory and the fetch stages as found in **Kyker**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to fetch as many instructions as possible to be waiting for processing at the earliest moment and avoid idle time (**Kyker**: column 1, lines 14-26).

Claim 42

Bhamidipati and **Kyker** disclosed the limitation wherein at least one of the storing and the retrieving step is performed independently of whether instructions are advancing between ones of the execution stages *(**Kyker**: column 1, lines 14-26; **Bhamidipati**: Figure 3, element 306)*.

Claim 43

Bhamidipati and **Kyker** disclosed the limitation wherein ones of the steps are repeated for multiple instructions *(**Bhamidipati**: column 1, line 15)*.

Claim 45

Bhamidipati did not explicitly state the method of Claim 40, and further including:

- providing an indication that one or more predetermined operations are occurring within one or more of the execution stages; and
- in response to the indication, allowing instructions to advance within the fetch stages.

Kyker demonstrated that it was known at the time of invention for pipelines to stall (column 3, lines 40-45). A stall prevents a current stage from moving forward when the next stage is not available. It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati** and **Kyker**'s pipeline in such a manner as an indication being provided when the execution stages have/are stalled/busy as suggested by **Kyker**'s own teaching. Clearly, **Bhamidipati**'s pipeline allows some of the fetch stages to continue (Figure 3, element 306) depending on the placement of the decoupling queue. It would have been obvious to one of ordinary skill in the art at the time of invention to provide a stall signal from the execution stages of **Bhamidipati** and **Kyker**'s pipeline and in response the fetch stages that are able to continue processing. This implementation would have been obvious because one of ordinary skill in the art would be motivated by the fact that stalls are common in pipeline technology and a decoupling queue is specifically designed to avoid stalls in the entire pipeline.

5. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Bhamidipati** et al. (USPN 6,112,295) in view of **Kyker** et al. (USPN 6,026,477) and in further view of **Alferness** et al. (USPN 5,577,259).

Bhamidipati and **Kyker** disclosed the limitation:

- wherein said first selection circuit includes a control circuit to allow an instruction to enter said pre-decode stage of processing while ... ones of the instructions are not advancing to a next stage of execution within said pipeline execution circuit (**Bhamidipati**: figure 3, element 306).

Bhamidipati and **Kyker** did not explicitly state the limitation

- wherein said pipeline execution circuit includes a microcode-controlled sequencer to control execution of extended stages of execution of extended-mode ones of the instructions, wherein during said extended stages of execution, ones of the instructions being executed by said pipeline execution circuit are not advancing to a next stage of execution within said pipeline execution circuit

Alferness demonstrated that it was known at the time of invention to utilize taught a microcode-controlled sequencer (column 1, lines 24-31) and the sequencer to control extended stages of execution of extended-mode instructions (column 1, lines 26-31; extended cycle instructions are the extended-mode instructions) wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit (column 4, lines 63-67). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati** and **Kyker**'s decoupling pipeline with extended-mode instruction handling ability which might stop the progression of the execution stages as found in **Alferness**' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a processor, which included the flexibility and control of micro-code over extended type instructions (**Alferness**: column 2, lines 38-42; column 3, lines 35-41).

6. Claims 26, 31 and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bhamidipati** et al. (USPN 6,112,295) in view **Alferness** et al. (USPN 5,577,259).

Claim 26

Bhamidipati did not explicitly state the pipeline circuit of Claim 21, wherein at least one of the execution stages includes a microcode-controlled sequencer to control execution of extended-mode instructions, and wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit. **Alferness** demonstrated that it was known at the time of invention to utilize taught a microcode-controlled sequencer (column 1, lines 24-31) and the sequencer to control extended stages of execution of extended-mode instructions (column 1, lines 26-31; extended cycle instructions are the extended-mode instructions) wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit (column 4, lines 63-67). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s decoupling pipeline with extended-mode instruction handling ability which might stop the progression of the execution stages as found in **Alferness**' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a processor, which included the flexibility and control of micro-code over extended type instructions (**Alferness**: column 2, lines 38-42; column 3, lines 35-41).

Claim 31

Bhamidipati did not explicitly state the pipeline circuit of Claim 27, wherein one of the execution stages includes a microcode sequencer to execute predetermined ones of the instructions in a manner that may temporarily affect movement of instructions within the execution stages. **Alferness** demonstrated that it was known at the time of invention to utilize taught a microcode-controlled sequencer (column 1, lines 24-31) and the sequencer to control extended stages of execution of extended-mode instructions (column 1, lines 26-31; extended cycle instructions are the extended-mode instructions) wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit (column 4, lines 63-67). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s decoupling pipeline with extended-mode instruction handling ability which might stop the progression of the execution stages as found in **Alferness**' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a processor, which included the flexibility and control of micro-code over extended type instructions (**Alferness**: column 2, lines 29-42; column 3, lines 35-41).

Claim 44

Bhamidipati did not explicitly state method of Claim 40, wherein one of the execution stages includes a microcode-controlled sequencer for executing extended-mode instructions, and further including executing one of the extended-mode instructions in a manner that temporarily delays the advancing of instructions between ones of the execution stages. **Alferness** demonstrated that it was known at the time of invention to utilize taught a microcode-controlled sequencer (column 1, lines 24-31) and the sequencer to control extended stages of execution of extended-mode instructions (column 1, lines 26-31; extended cycle instructions are the extended-mode instructions) wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit (column 4, lines 63-67). It would have been obvious to one of ordinary skill in the art at the time of invention to implement **Bhamidipati**'s decoupling pipeline with extended-mode instruction handling ability which might stop the progression of the execution stages as found in **Alferness**' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a processor, which included the flexibility and control of micro-code over extended type instructions (**Alferness**: column 2, lines 29-42; column 3, lines 35-41).

7. Claims 1, 3-4, 21, 27, 32, 39, 40 and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bhamidipati et al. (USPN 6,112,295) in view of Hayes, John P., "Computer Architecture and Organization". (Previous rejection maintained).

In regard to claim 1, Bhamidipati disclosed the limitations:

- *For use in an instruction processor that executes instructions included in a predetermined instruction set* (column 1, lines 51-53), *comprising:*
  - *a pipeline execution circuit to process a first predetermined number of instructions simultaneously* (Figure 1, element 106), *each of said first predetermined number of instructions being in a respectively different stage of execution within said pipeline execution circuit* (Figure 2)*; and*
  - *a pipeline fetch circuit coupled to provide each of the first predetermined number of instructions to said pipeline execution circuit* (Figure 1, element 106), *the pipeline fetch circuit to retain a second predetermined number of instructions simultaneously* (Figure 3, elements 300, 302, 304 and 308), *each of said second predetermined number of instructions being in a respectively different stage of processing within said pipeline fetch circuit*

(Figure 2), *an instruction being capable of advancing to a next stage of execution within said pipeline fetch circuit independently of the times at which instructions advance to a next stage of execution within said pipeline execution circuit* (Figure 3, element 306)

Bhamidipati did explicitly state a synchronous pipeline. However, Bhamidipati indicated that it was known at the time of invention to process instructions in a synchronous pipeline (column 2, lines 43-55; and Figures 1-2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's decoupling queue system in a synchronous pipeline as found in Bhamidipati's own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a pipeline as close to conventional as possible (easier support in the future). Furthermore, Bhamidipati's teachings suggest this is the desired implementation by discussing this type of pipeline as if it is the type being modified by Bhamidipati's improvement (column 2, lines 43-55; Figures 1-2) and by the great lengths Bhamidipati takes to keep the improvements within a single clock cycle (column 1, lines 50-61).

Bhamidipati did not explicitly state the fetch circuit to transfer each instruction processed by the fetch circuit directly from one of the fetch stages to one of the execution stages. Hayes demonstrated that it was known at the time of invention to directly couple a fetch stage to an execution stage (page 223-224, Figures 3.68-3.69) in a typical pipeline arrangement. It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's pipeline as a fetch stage directly coupled to an execution stage as found in Hayes' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to implement a standard (and thus well known/easy to implement) pipeline and Bhamidipati further indicated many different types pipeline stages can be used to implement the invention (column 3, lines 54-64).

Claims 3-4

Please see rejections under paragraph 2.

In regard to claim 21, Bhamidipati disclosed the limitations:

- *For use in an instruction processor* (column 1, lines 51-53), *comprising:*
  - *an execution circuit to provide a first predetermined number of execution stages* (Figure 1, element 106), *each being capable of performing a respective processing operation on a respective instruction* (execution stages inherently process instructions)*; and*
  - *a fetch circuit coupled to the execution circuit to provide a second predetermined number of fetch stages* (Figure 3, elements 300, 302, 304 and 308), *each fetch stage being capable of performing a respective pre-execution operation on a respective instruction* (elements of Figure 3 show operations that are prior to execution), *ones of the instructions processed within the fetch stages being*

capable of advancing to different available fetch stages independently of whether instructions are advancing within the execution stages (Figure 3, element 306).

Bhamidipati did explicitly state a synchronous pipeline. However, Bhamidipati indicated that it was known at the time of invention to process instructions in a synchronous pipeline (column 2, lines 43-55; and Figures 1-2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's decoupling queue system in a synchronous pipeline as found in Bhamidipati's own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a pipeline as close to conventional as possible (easier support in the future). Furthermore, Bhamidipati's teachings suggest this is the desired implementation by discussing this type of pipeline as if it is the type being modified by Bhamidipati's improvement (column 2, lines 43-55; Figures 1-2) and by the great lengths Bhamidipati takes to keep the improvements within a single clock cycle (column 1, lines 50-61).

Bhamidipati did not explicitly state *the fetch circuit to transfer each instruction processed by the fetch circuit directly from one of the fetch stages to one of the execution stages*. Hayes demonstrated that it was known at the time of invention to directly couple a fetch stage to an execution stage (page 223-224, Figures 3.68-3.69) in a typical pipeline arrangement. It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's pipeline as a fetch stage directly coupled to an execution stage as found in Hayes' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to implement a standard (and thus well known/easy to implement) pipeline and Bhamidipati further indicated many different types pipeline stages can be used to implement the invention (column 3, lines 54-64).

In regard to claim 27, Bhamidipati disclosed the limitations:

- *A instruction pipeline circuit for processing instructions within a data processing system* (column 1, lines 51-53), *comprising:*
  - *a first predetermined number of fetch stages to simultaneously process at least a first predetermined number of instructions* (Figure 3, elements 300, 302, 304 and 308)*;*
  - *a second predetermined number of execution stages to simultaneous process a second predetermined number of instructions* (Figure 1, element 106)*; and*
  - *wherein at least one of the fetch stages is capable of providing an instruction to a different one of the fetch stages that is ready to receive an instruction irrespective of movement of instructions between the execution stages* (Figure 3, element 306).

Bhamidipati did explicitly state a synchronous pipeline. However, Bhamidipati indicated that it was known at the time of invention to process instructions in a synchronous pipeline (column 2, lines 43-55; and Figures 1-2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's decoupling queue system in a synchronous pipeline as found in Bhamidipati's own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a pipeline as close to conventional as possible (easier support in the future). Furthermore, Bhamidipati's teachings suggest this is the desired implementation by discussing this type of pipeline as if it is the type being modified by Bhamidipati's improvement (column 2, lines 43-55; Figures 1-2) and by the great lengths Bhamidipati takes to keep the improvements within a single clock cycle (column 1, lines 50-61).

Bhamidipati did not explicitly state *the fetch circuit to transfer each instruction processed by the fetch circuit directly from one of the fetch stages to one of the execution stages*. Hayes demonstrated that it was known at the time of invention to directly couple a fetch stage to an execution stage (page 223-224, Figures 3.68-3.69) in a typical pipeline arrangement. It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's pipeline as a fetch stage directly coupled to an execution stage as found in Hayes' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to implement a standard (and thus well known/easy to implement) pipeline and Bhamidipati further indicated many different types pipeline stages can be used to implement the invention (column 3, lines 54-64).

In regard to claim 32, Bhamidipati disclosed the limitations:

- *A instruction pipeline to execute instructions* (column 1, lines 51-53), *comprising:*
  - *an execution circuit having a first predetermined number of execution stages to execute a first predetermined number of instructions substantially simultaneously* (Figure 1, element 106)*; and*
  - *a fetch circuit having a second predetermined number of fetch stages to perform pre-execution operations on at least a second predetermined number of instructions substantially simultaneously* (Figure 3, elements 300, 302, 304 and 308), *at least one of the at least second predetermined number of instructions being capable of advancing between different ones of the fetch stages regardless of whether an instruction is being transferred by the fetch circuit to the execution circuit* (Figure 3, element 306).

Bhamidipati did explicitly state a synchronous pipeline. However, Bhamidipati indicated that it was known at the time of invention to process instructions in a synchronous pipeline (column 2, lines 43-55; and Figures 1-2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's decoupling queue system in a synchronous pipeline as found in Bhamidipati's own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a pipeline as close to conventional as possible (easier support in the future). Furthermore, Bhamidipati's teachings suggest this is the desired implementation by discussing this type of pipeline as if it is the type being modified by Bhamidipati's improvement (column 2, lines 43-55; Figures 1-2) and by the great lengths Bhamidipati takes to keep the improvements within a single clock cycle (column 1, lines 50-61).

Bhamidipati did not explicitly state *the fetch circuit to transfer each instruction processed by the fetch circuit directly from one of the fetch stages to one of the execution stages.* Hayes demonstrated that it was known at the time of invention to directly couple a fetch stage to an execution stage (page 223-224, Figures 3.68-3.69) in a typical pipeline arrangement. It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's pipeline as a fetch stage directly coupled to an execution stage as found in Hayes' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to implement a standard (and thus well known/easy to implement) pipeline and Bhamidipati further indicated many different types pipeline stages can be used to implement the invention (column 3, lines 54-64).

In regard to claim 39, Bhamidipati and Hayes disclosed the limitation *wherein the fetch circuit includes a circuit that allows instructions to advance within the* *second predetermined number of fetch stages if one of the execution stages is performing a predetermined function* (Bhamidipati: Figure 3, element 306).

In regard to claim 40, Bhamidipati disclosed the limitations:

- *A method of processing instructions within a pipeline of an instruction processor* (column 1, lines 51-53), *comprising:*
  - a.) performing pre-execution operations on a first predetermined number of instructions substantially simultaneously within a first predetermined number of fetch stages of the pipeline *(Figure 3, elements 300, 302, 304 and 308);*
  - *b.) executing a second predetermined number of instructions substantially simultaneously within a second predetermined number of execution stages of the pipeline (*Figure 1, element 106*); and*
  - c.) allowing one or more of the first predetermined number of instructions to advance between ones of the fetch stages independently of whether any of the second predetermined number of instructions are advancing between ones of the execution stages *(*Figure 3, element 306*).*

Bhamidipati did explicitly state a synchronous pipeline. However, Bhamidipati indicated that it was known at the time of invention to process instructions in a synchronous pipeline (column 2, lines 43-55; and Figures 1-2). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's decoupling queue system in a synchronous pipeline as found in Bhamidipati's own teachings. This implementation would have been obvious because one of ordinary skill in the art would be motivated to design a pipeline as close to conventional as possible (easier support in the future). Furthermore, Bhamidipati's teachings suggest this is the desired implementation by discussing this type of pipeline as if it is the type being modified by Bhamidipati's improvement (column 2, lines 43-55; Figures 1-2) and by the great lengths Bhamidipati takes to keep the improvements within a single clock cycle (column 1, lines 50-61).

Bhamidipati did not explicitly state *wherein each of the second predetermined number of instructions were received directly from one of the fetch stages.* Hayes demonstrated that it was known at the time of invention to directly couple a fetch stage to an execution stage (page 223-224, Figures 3.68-3.69) in a typical pipeline arrangement. It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's pipeline as a fetch stage directly coupled to an execution stage as found in Hayes' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to implement a standard (and thus well known/easy to implement) pipeline and Bhamidipati further indicated many different types pipeline stages can be used to implement the invention (column 3, lines 54-64).

In regard to claim 46, Bhamidipati disclosed the limitations:

- *A pipeline circuit for use in an instruction processor* (column 1, lines 51-53), *comprising:*
  - *instruction fetch means for performing pre-execution operations on a first predetermined number of instructions substantially simultaneously within a first predetermined number of fetch stages* (Figure 3, elements 300, 302, 304 and 308);
  - *instruction execution means for executing a second predetermined number of instructions substantially simultaneously within a second predetermined number of execution stages* (Figure 1, element 106); *and*
  - *wherein the instruction fetch means includes means for allowing at least one of the first predetermined number of instructions to advance within the fetch stages irrespective of whether instructions are advancing within the execution stages* (Figure 3, element 306).

Bhamidipati did not explicitly state *wherein each of the second predetermined number of instructions were received directly from one of the fetch stages*. Hayes demonstrated that it was known at the time of invention to directly couple a fetch stage to an execution stage (page 223-224, Figures 3.68-3.69) in a typical pipeline arrangement. It would have been obvious to one of ordinary skill in the art at the time of invention to implement Bhamidipati's pipeline as a fetch stage directly coupled to an execution stage as found in Hayes' teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to implement a standard (and thus well known/easy to implement) pipeline and Bhamidipati further indicated many different types pipeline stages can be used to implement the invention (column 3, lines 54-64).

8. Claims 2, 5-6, 22-25, 28-30, 33-38, 41-43 and 45 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bhamidipati** et al. (USPN 6,112,295) in view of **Hayes**, John P., “Computer Architecture and Organization” and in further view of **Kyker** et al. (USPN 6,026,477). (Previous rejection maintained). Please see above rejections of claims under paragraphs 2 and 4.

9. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Bhamidipati** et al. (USPN 6,112,295) in view of **Hayes**, John P., “Computer Architecture and Organization” in view of **Kyker** et al. (USPN 6,026,477) and in further view of **Alferness** et al. (USPN 5,577,259). (Previous rejection maintained). Please see above rejections of claims under paragraph 5.

10. Claims 26, 31 and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Bhamidipati** et al. (USPN 6,112,295) in view of **Hayes**, John P., “Computer Architecture and Organization” and in further view of **Alferness** et al. (USPN 5,577,259). (Previous rejection maintained). Please see above rejections of claims under paragraph 6.

Response to Arguments

11. Applicant's arguments filed 19 July 2004 have been fully considered but they are not persuasive. Applicant argued, with previously cited reasons for rejection (maintained in this Office Action): 1) **Bhamidipati** does not provide fetch circuit independent of execution circuit (Brief: page 13, bottom); 2) **Bhamidipati** does not disclose each instruction retained within fetch circuit within a respectively different stage of processing (Brief: page 14-15 and 21); 3) **Hayes** fails to provide for **Bhamidipati**'s alleged deficiencies (Brief: page 15-16); 4) no motivation to combine **Bhamidipati** and **Hayes** (Brief: page 16-17 and page 22-24); 5) **Bhamidipati** does not provide for direct coupling of fetch stages and execution stages (Brief: page 18, middle); and 6) no motivation to combine with **Alferness** (pages 26 and 27, middle).

First, **Bhamidipati**'s queue provides independence between fetch and execution circuits. The fetch circuit can operate regardless of what the execution circuit is doing by utilizing the queue.

Second, "stage of processing" is extremely broad. A queue can provide for a stage of processing in the sense that the instruction is at least stored/retained in a queue (processes). One instruction in the queue provides each instruction in respectively different stages.

Third, **Hayes** clearly demonstrated a well known pipeline configuration and was cited for that specific reason.

Fourth, **Hayes** and **Bhamidipati** are motivated as previously indicated. **Bhamidipati** itself suggests multiple differing configurations. It is unclear from the cited references that **Bhamidipati** is improving **Hayes**. Additionally, this is irrelevant as **Hayes** was cited for a specifically know configuration of a pipeline. Finally, Applicant states, "those skilled in the art of modern data processing systems are rarely motivated to turn the clock back more than two decades ...". Yet, for example, the automobile (modern technology) uses the wheel (ancient technology). Thus, the combination is maintained.

Fifth, direct coupling clearly provided (see **Bhamidipati**: figure 1 and column 3, line 58, considering queue part of "fetch circuit").

Sixth, note above rejections, micro-code instructions provide flexibility.

Applicant's raised issues having been addressed, the rejections are maintained and a new rejection using similar art is added.

Conclusion

12. This Office Action is non-Final.

Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to William H. Wood whose telephone number is (703)305-3305. The examiner can normally be reached 7:30am - 5:00pm Monday thru Thursday and 7:30am - 4:00pm every other Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662. The fax phone numbers for the organization where this application or proceeding is assigned are (703)746-7239 for regular communications and (703)746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703)305-3900.

William H. Wood
October 12, 2004

[illegible]

ANIL KHATRI
PRIMARY EXAMINER